2. Account Compromised

What is the name of the compromised account

```
* Tips:
* Analyse last 1000 line of logs
* Check if the user had multiple unsuccessful break in and then success
this will be mostly compromised account
* Check for Suspected Activity
```

```
┌──(imenⓀhbtn-lab)-
[…/web_application_security/0x0c_web_application_foresics]
└─$ ./2-accounts.sh
root
```

Command Breakdown:

```
tail -n 1000 auth.log | grep -E "root" | sort | uniq -c | sort -nr | head -n
1 | awk '{print $12}'
```

1. **`tail -n 1000 auth.log`**:
   - This retrieves the last **1000 lines** from the `auth.log` file. The `auth.log` file typically contains authentication logs, such as login attempts and sudo commands.
   - `tail` is used to view the most recent log entries, and the `-n 1000` option ensures that only the last 1000 lines are included in the output.
2. **`grep -E "root"`**:
   - This filters the last 1000 lines of the `auth.log` file to only include lines that contain the string `root`. It looks for any log entries involving the user `root`. The `-E` option allows for extended regular expressions, though it's not strictly necessary here unless you're planning to use more complex patterns later.
3. **`sort`**:
   - This sorts the filtered lines alphabetically. Sorting is necessary before using `uniq -c`, which will count occurrences of identical lines.
4. **`uniq -c`**:
   - This command counts the number of occurrences of each unique line in the sorted output. Essentially, it will count how many times each `root`-related log entry appears in the filtered `auth.log`.

5. `sort -nr`:
    - This sorts the counted occurrences in **numerical reverse order**, so that the log entries with the highest count will appear first.
6. `head -n 1`:
    - This command then takes the top (most frequent) log entry from the sorted list. This ensures that only the most frequent entry involving `root` is passed along.
7. `awk '{print $12}'`:
    - Finally, `awk` is used to print the **12th field** from the top log entry. This assumes that the 12th field contains some relevant information related to the `root` authentication event. This could be an IP address, a command, or another piece of data depending on the log format.

What the command does:

This command pipeline retrieves the last 1000 lines from the `auth.log` file, filters for entries involving `root`, counts the occurrences of each unique log entry, sorts them by frequency, and then extracts the 12th field from the most frequent `root`-related log entry.

Example scenario:

If the `auth.log` file contains entries like:

```
Feb 24 14:23:56 server sshd[2345]: Accepted password for root from
192.168.1.10 port 22
Feb 24 14:23:59 server sshd[2345]: Accepted password for root from
192.168.1.11 port 22
Feb 24 14:24:05 server sshd[2345]: Failed password for root from
192.168.1.10 port 22
```

And the most frequent log entry related to `root` looks like this:

```
  10 Feb 24 14:23:56 server sshd[2345]: Accepted password for root from
192.168.1.10 port 22
```

After running the command, the output will display:

```
192.168.1.10
```

This indicates that the IP address `192.168.1.10` was involved in the most frequent `root` authentication event in the last 1000 lines of the `auth.log` file.

Use case:

This command is useful for quickly identifying the most common log entry related to `root`, such as the most frequent IP address or event (e.g., login attempt) in the `auth.log`. It can help detect patterns in

authentication or pinpoint the source of the most frequent `root` access.